

Laboratorio 2: Memoria Virtual vs Física y Caché y Rendimiento

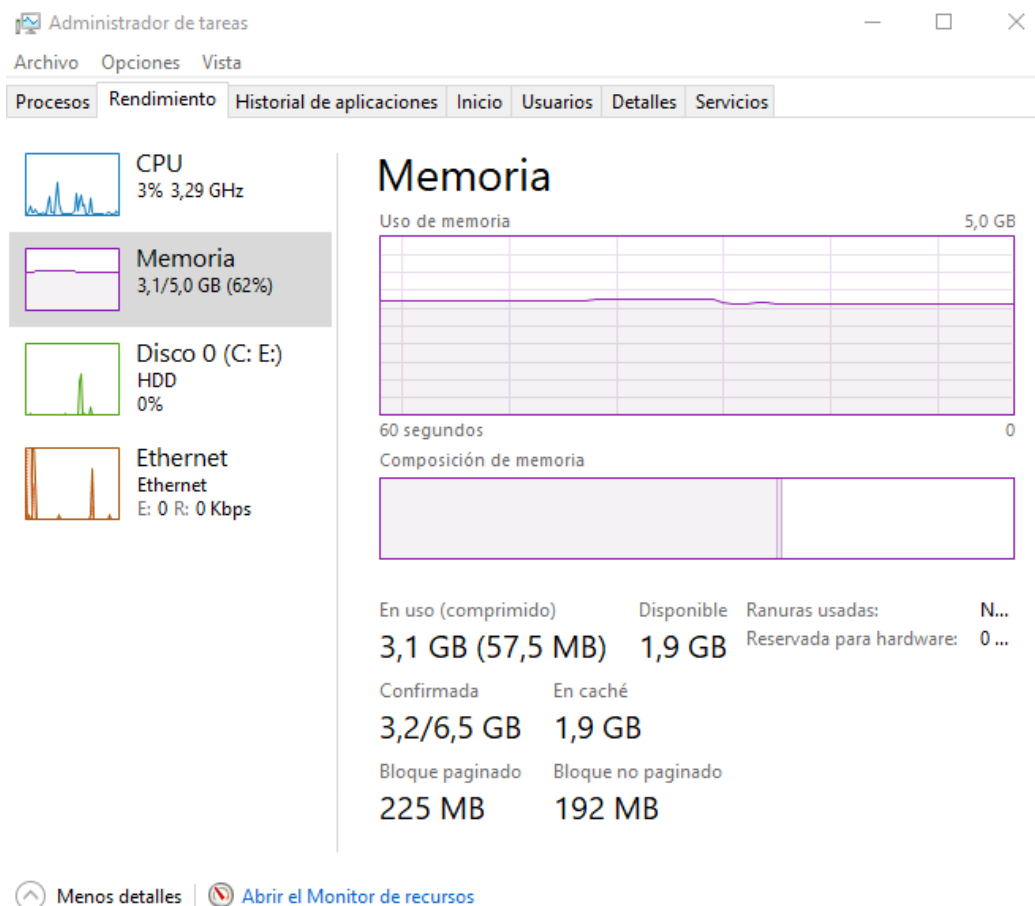
Estados de Memoria Virtual y Física

Durante la prueba, se monitoreó el uso de la memoria RAM y de la memoria virtual usando el Administrador de tareas de Windows.

Se ejecutaron programas que consumen gran cantidad de memoria para observar cómo el sistema operativo usa la memoria virtual cuando la RAM se acerca a su límite.

Observaciones:

- La memoria física disponible era aproximadamente 5 GB.
- En un momento, la memoria total usada (RAM + virtual) alcanzó cerca de 6.5 GB, confirmando que se estaba usando memoria virtual para suplir la falta de RAM.
- Esto se pudo ver claramente en el Administrador de tareas con la columna “Memoria comprometida” y el gráfico de uso de memoria.





```
import time
```

```
print("Inicio de consumo de memoria (versión para 5 GB de RAM)...")
```

```
datos = []
```

```
try:
```

```
    for i in range(250): # Total: ~2.5 GB (250 * 10MB)
```

```
        datos.append('X' * 10**7) # 10 MB por iteración
```

```
        print(f"Memoria acumulada: {round(len(datos) * 10 / 1024, 2)} GB")
```

```
        time.sleep(0.2)
```

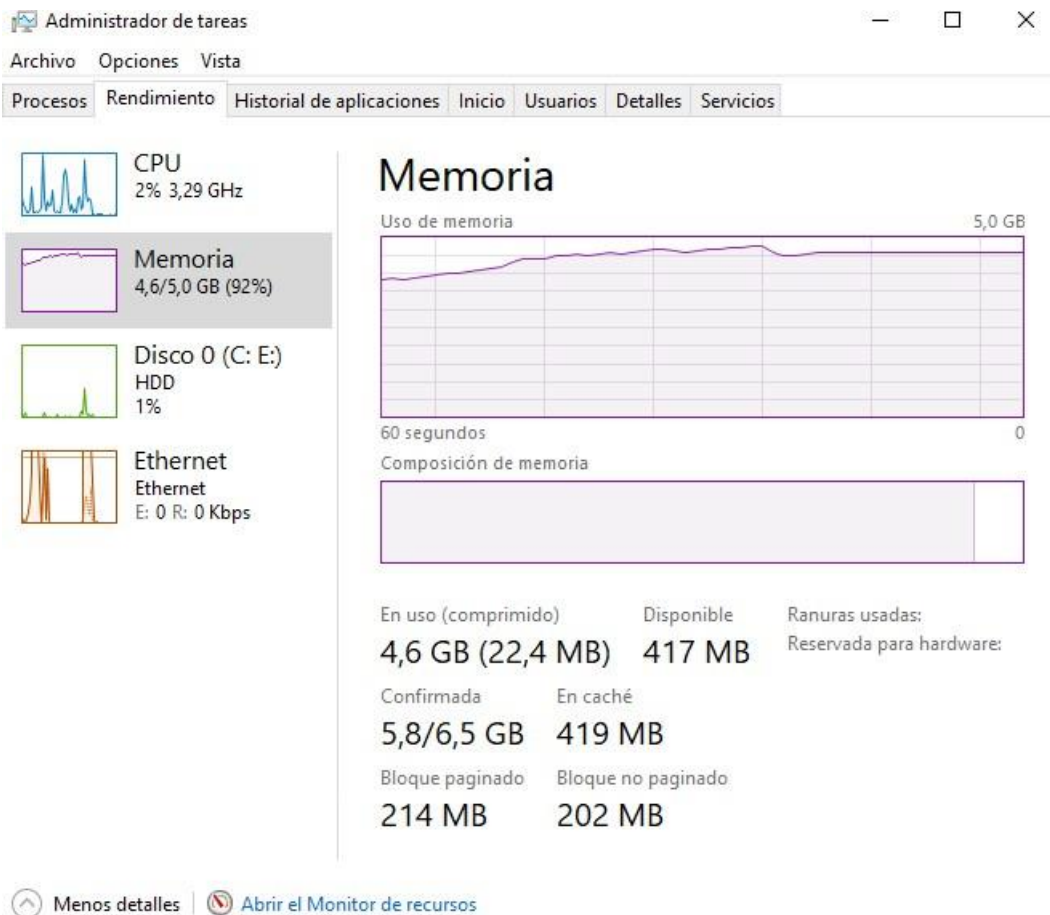
```
except MemoryError:
```

```
    print("¡Se alcanzó el límite de memoria!")
```

```
print("Fin del programa.")
```

```
input("Presiona Enter para salir...")|
```

```
Memoria acumulada: 1.15 GB
Memoria acumulada: 1.16 GB
Memoria acumulada: 1.17 GB
Memoria acumulada: 1.18 GB
Memoria acumulada: 1.19 GB
Memoria acumulada: 1.2 GB
Memoria acumulada: 1.21 GB
Memoria acumulada: 1.22 GB
Memoria acumulada: 1.23 GB
Memoria acumulada: 1.24 GB
Memoria acumulada: 1.25 GB
Memoria acumulada: 1.26 GB
Memoria acumulada: 1.27 GB
Memoria acumulada: 1.28 GB
Memoria acumulada: 1.29 GB
Memoria acumulada: 1.3 GB
Memoria acumulada: 1.31 GB
Memoria acumulada: 1.32 GB
Memoria acumulada: 1.33 GB
Memoria acumulada: 1.34 GB
Memoria acumulada: 1.35 GB
Memoria acumulada: 1.36 GB
Memoria acumulada: 1.37 GB
Memoria acumulada: 1.38 GB
Memoria acumulada: 1.39 GB
Memoria acumulada: 1.4 GB
Memoria acumulada: 1.41 GB
Memoria acumulada: 1.42 GB
Memoria acumulada: 1.43 GB
```



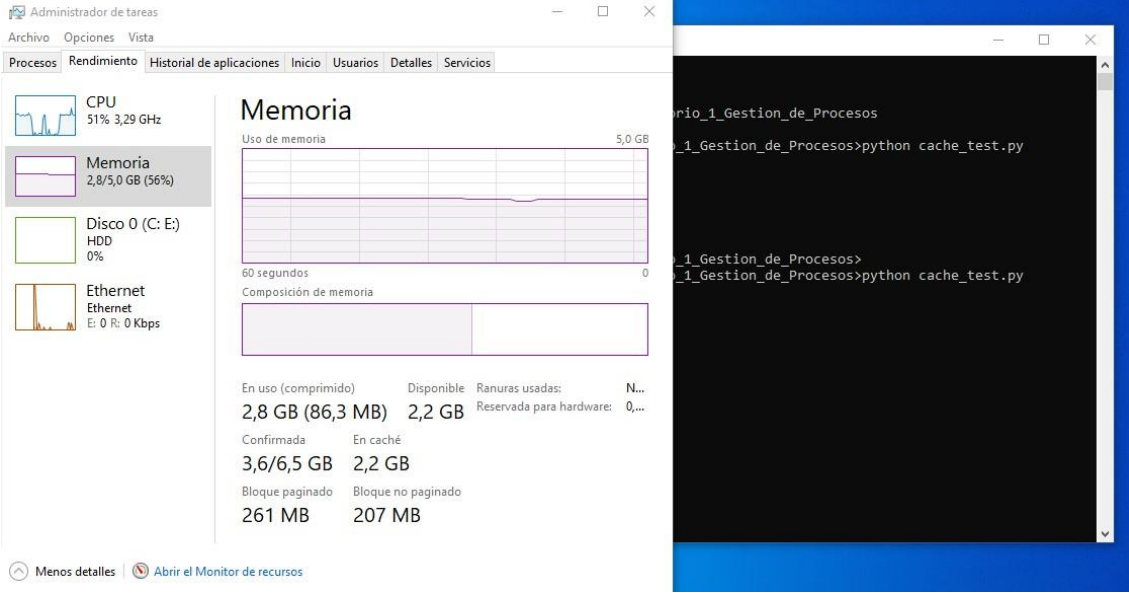
Prueba de Caché y Rendimiento

Se utilizó un programa en Python que realiza dos tipos de acceso a un arreglo grande de datos:

- **Acceso secuencial:** recorre los datos en orden, aprovechando la caché de CPU.
- **Acceso aleatorio:** accede a posiciones aleatorias, lo que genera un peor aprovechamiento de la caché.

Se ejecutó la prueba antes y después de reiniciar el equipo para “limpiar” la caché y observar diferencias en los tiempos de ejecución.

```
C:\Users\Eduardo>pip install numpy
Collecting numpy
  Downloading numpy-2.3.0-cp313-cp313-win_amd64.whl.metadata (60 kB)
  Downloading numpy-2.3.0-cp313-cp313-win_amd64.whl (12.7 MB)
----- 12.7/12.7 MB 6.8 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-2.3.0
```



The screenshot shows the Windows Task Manager Performance tab. The Memory section indicates 56% usage (2.8 GB of 5.0 GB). The Memory Usage graph shows a steady increase over 60 seconds. The Memory Composition table shows 2.8 GB in use (86.3 MB compressed), 2.2 GB available, 3.6/6.5 GB confirmed, and 261 MB paged. The Memory Composition table also shows 2.2 GB in cache, 207 MB paged, and 207 MB not paged. The terminal window shows the execution of a Python script named cache_test.py.

```
cache_test: Bloc de notas
Archivo Edición Formato Ver Ayuda

import time
import numpy as np

print("Inicio prueba de caché y rendimiento")

size = 10**7 # Tamaño grande para afectar caché
array = np.arange(size)

start = time.time()

# Acceso secuencial: buen uso de caché
sum_seq = 0
for i in range(size):
    sum_seq += array[i]

end_seq = time.time()
print(f"Acceso secuencial: {end_seq - start:.4f} segundos")

start = time.time()

# Acceso aleatorio: peor uso de caché
import random
sum_rand = 0
for _ in range(size):
    sum_rand += array[random.randint(0, size-1)]

end_rand = time.time()
print(f"Acceso aleatorio: {end_rand - start:.4f} segundos")

print("Fin prueba de caché y rendimiento")
input("Presiona Enter para cerrar...")
```

Con el cache sucio:

```
C:\Users\Eduardo\Desktop\Laboratorios_SO_Windows_Eduardo\Laboratorio_1_Gestion_de_Procesos>python cache_test.py
Inicio prueba de caché y rendimiento
Acceso secuencial: 3.5443 segundos
Acceso aleatorio: 15.6272 segundos
Fin prueba de caché y rendimiento
Presiona Enter para cerrar...
```

Con el cache limpio:

```
C:\Users\Eduardo\Desktop\Laboratorios_SO_Windows_Eduardo\Laboratorio_1_Gestion_de_Procesos>python cache_test.py
Inicio prueba de caché y rendimiento
Acceso secuencial: 3.3430 segundos
Acceso aleatorio: 13.5655 segundos
Fin prueba de caché y rendimiento
Presiona Enter para cerrar...
```

Análisis

- El sistema operativo usa memoria virtual cuando la RAM está saturada para mantener el funcionamiento de los programas.
- La diferencia en los tiempos de acceso secuencial y aleatorio refleja el impacto de la memoria caché en el rendimiento.
- Reiniciar la PC permitió limpiar la caché, lo que se tradujo en una ligera mejora en los tiempos de ejecución, especialmente en el acceso aleatorio.
- La CPU alcanzó hasta un 51% de uso durante la prueba, confirmando la carga generada.

Conclusión

Las pruebas mostraron claramente cómo funciona la memoria en el sistema operativo y la importancia de la caché para optimizar el rendimiento.

- La memoria virtual es esencial para ampliar la capacidad de memoria cuando la física se agota, evitando bloqueos.
- La caché mejora notablemente el rendimiento en accesos secuenciales, mientras que en accesos aleatorios su beneficio disminuye.
- Reiniciar el sistema es una forma efectiva de limpiar la caché y mejorar temporalmente el rendimiento.